Atividade 1: Retângulo

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

- Alterar o **solucao-1.js**
- Obter, via teclado, os valores para cálculo do:
  - Perímetro do retângulo
  - Área do retângulo
- Utilize o módulo **readline**.
- Enviar arquivo compactado no formato **zip** para angela.de.a.ferreira@accenture.com
- Prazo: 23/10/2023

exemplo-readline.js

```
const readln = require("readline");

const readline = readln.createInterface({
  input: process.stdin,
  output: process.stdout,
});

readline.question("Qual o seu nome? ", (name) => {
  console.log(`Ola ${name}!`);
  readline.close();
});
```

Solucao-1.js

```
var rect = require('./retangulo-1.js')

function solveRect(l,b) {
    console.log("Solução para o retangulo com l = " + l + " e b =" + b);

    if (l < 0 || b < 0){
        console.log("as dimensoes do retangulo devem ser maior que zero: l= " +
l + ", e b+ " +b);
    }
    else{
        console.log("A area do retangulo com dimensoes comprimento = " + l + "
e largura = " + b + " e " + rect.area(l,b));
        console.log("O perimetro do retangulo com dimensoes comprimento = " + l
+ " e largura = " + b + " e " + rect.perimeter(l,b));

    }
}
//solveRect(2,4);
//solveRect(3,5);
solveRect(-3,5);
```

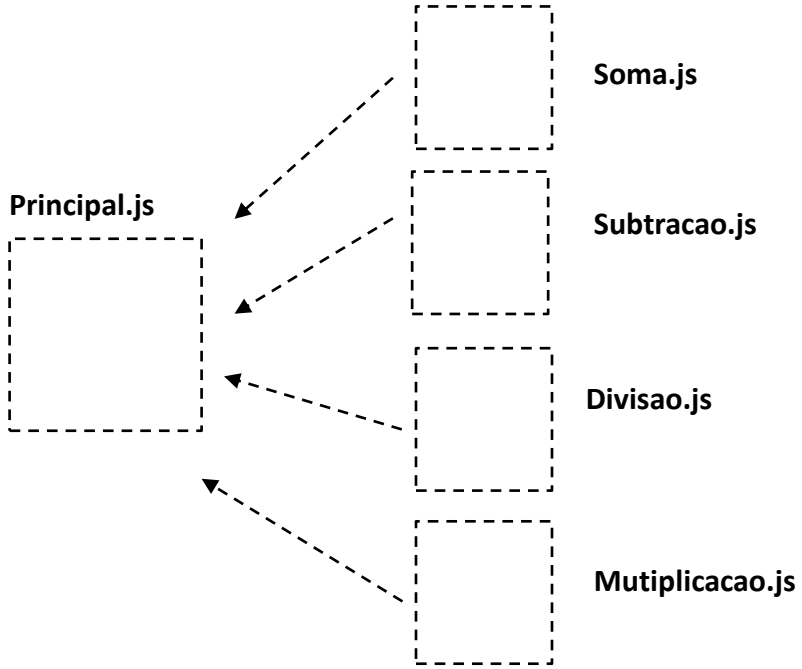
Retangulo-1.js

```
exports.perimeter = function(x,y){
    return (2*(x+y));
},

exports.area = function(x,y){
    return(x*y)
}
```

Atividade 2: Calculadora

- Crie uma calculadora.
- Ela deve ter 5 arquivos.
- Sendo 4 com as operações básicas e 1 com a função principal.
- Usar o **prompt-sync**
- Enviar arquivo compactado (sem a pasta **node_module**) no formato **zip** para angela.de.a.ferreira@accenture.com
- Prazo: 23/10/2023



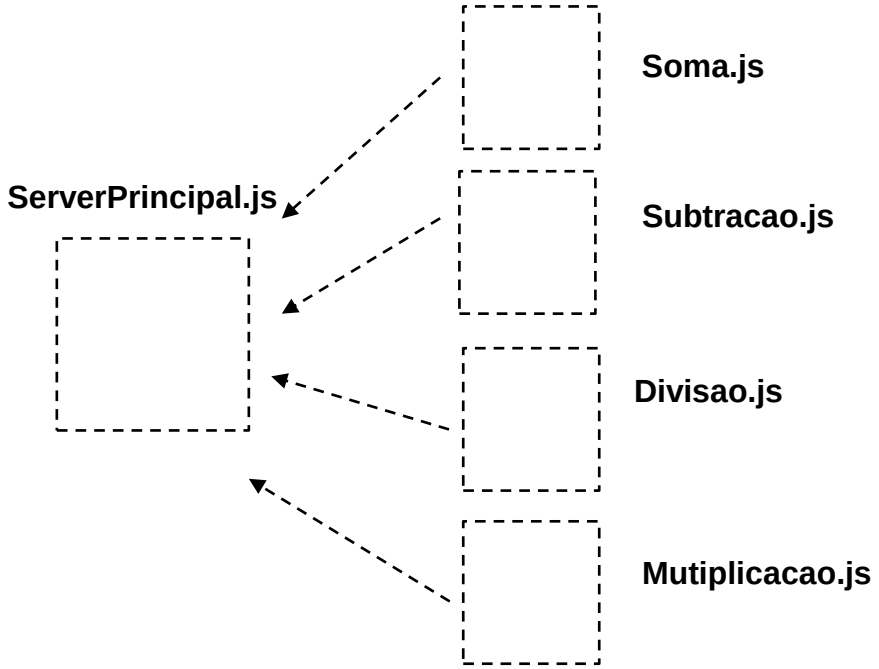
exemplo-promp-sync.js

```
const prompt = require('prompt-sync')();

const num = prompt('Informe um número: ');
console.log('Seu número + 4 =');
console.log(Number(num) + 4);
```

Atividade 3: API Calculadora

- Crie uma API Calculadora.
- A API deverá ter:
  - Pelo menos 4 end-points do tipo GET.
  - Usar os arquivos externos com as 4 operações básicas.
  - O Front-End deve enviar para o servidor Node os números e o tipo de operação que será realizada (opcional).
- Enviar arquivo compactado (sem a pasta **node_module**) no formato **zip** para angela.de.a.ferreira@accenture.com
- Prazo: 23/10/2023

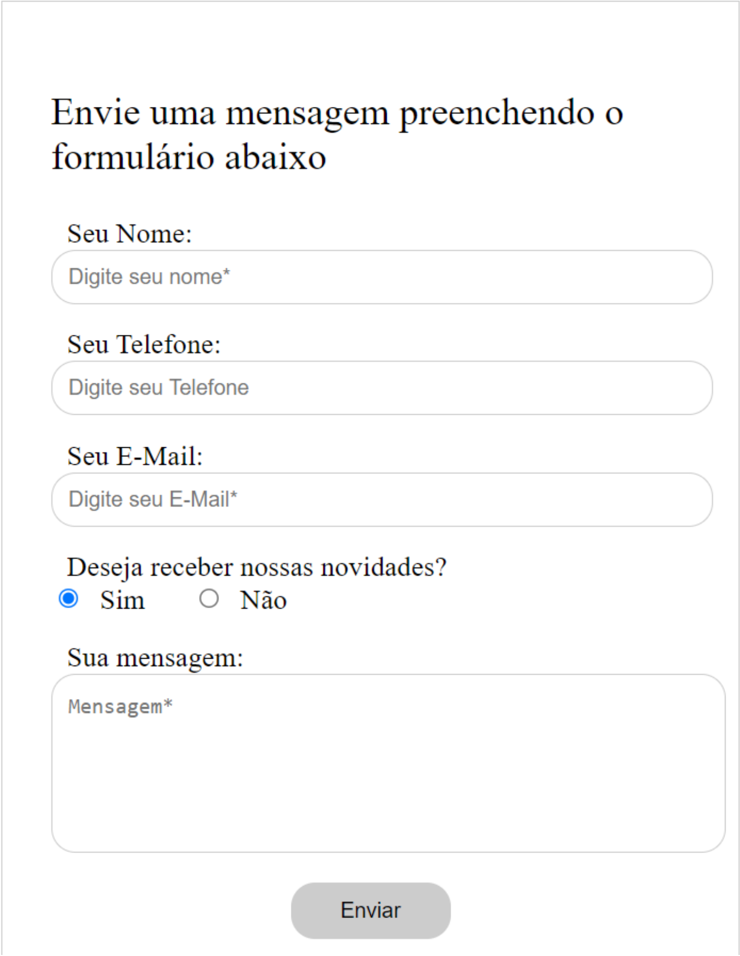


Atividade 4: MySQL – Passagem de valor

- Criar uma tabela usuários no MySql.

```
CREATE TABLE `user` (
  `id` smallint unsigned NOT NULL AUTO_INCREMENT,
  `nome` varchar(200) NOT NULL,
  `telefone` varchar(15) DEFAULT NULL,
  `email` varchar(150) DEFAULT NULL,
  `novidades` tinyint(1) NOT NULL,
  `mensagem` text,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

- Criar a API que vai receber os dados enviados do formulário.
- Salvar os dados na tabela.
  - PassagemDeValores.html
  - Node_Sparametro2.js
- Enviar arquivo compactado (sem a pasta **node_module**) no formato **zip** para angela.de.a.ferreira@accenture.com
- Prazo: 25/10/2023



Atividade 5: AXIOS ViaCEP

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

Crie uma API axios para consumir a API VIACEP.

- Criar uma pagina web simples para informar o CEP.
- Criar a tabela de CEP no mysql.
- Fazer GET na viacep.
- Recuperar os dados.
- Conectar no mySQL.
- Salvar em uma tabela o cep recuperado.
- https://viacep.com.br/ws/01001000/json/

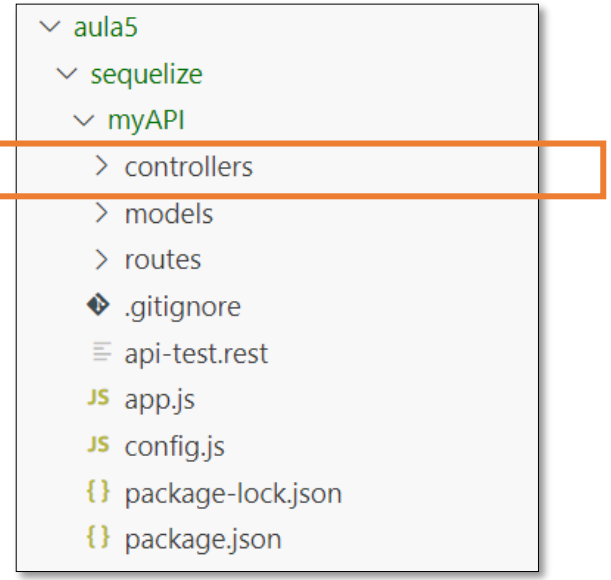
```
CREATE TABLE IF NOT EXISTS `cep` (
    `cep` VARCHAR(9) PRIMARY KEY,
    `logradouro` VARCHAR(300) NOT NULL,
    `complemento` VARCHAR(200) NULL,
    `bairro` VARCHAR(200) NOT NULL,
    `localidade` VARCHAR(200) NOT NULL,
    `uf` VARCHAR(2) NOT NULL,
    `ibge` VARCHAR(10) NOT NULL,
    `gia` VARCHAR(200) NULL,
    `ddd` VARCHAR(3) NOT NULL,
    `siafi` VARCHAR(10) NOT NULL
) ENGINE=InnoDB;
```

Atividade 6: ORM Sequelize - myAPI

- Colocar as rotas de GET e PUT da API **myAPI** funcionarem adequadamente.
- Para isso implemente, no userController.js, as funções de **getUsers** e **updateUsers**.
- Use como base a implementação do **deleteUser**.

aula5 > sequelize > myAPI > controllers > JS userController.js > ...

```
15
16  // Read (GET)
17  const getUsers = async (req, res) => {
18    // Implementar aqui
19    res.status(200).json({ mensagem: "Não implementado!" })
20  };
21
22  // Update (PUT)
23  const updateUser = async (req, res) => {
24    // Implementar aqui
25    res.status(200).json({ mensagem: "Não implementado!" })
26  };
27
```



Atividade 7: Swagger - myAPI

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657

- Adicionar o **Swagger** na API do Sequelize – **myApi.**
- Segui o passo a passo descrito no arquivo **README-Atividade.md** localizado dentro da pasta **/aula6/swagger/myAPI-com-Swagger**
- Enviar evidências por e-mail com a tela do Swagger funcionando

Preview README-Atividade.md

Atividade: Instalação do Swagger no Node.js

1. Instalação de Dependências

Instalar a biblioteca swagger-jsdoc e swagger-ui-express para criar a documentação do Swagger e fornecer um UI interativo para visualizá-la:

```
npm install swagger-jsdoc@5.0.1 --save-exact
npm install swagger-ui-express --save
```

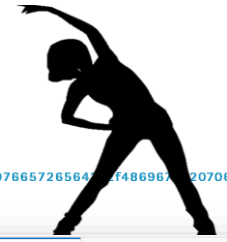
2. Configuração do Swagger

Crie um arquivo chamado **swagger.js** na raiz do projeto para configurar o Swagger.

```
const swaggerJSDoc = require('swagger-jsdoc');

const options = {
  definition: {
    openapi: '3.0.0',
    info: {
      title: 'API do Sequelize', // Título da sua API
      version: '1.0.0', // Versão da sua API
      description: 'Documentação da API do Sequelize', // Descrição da sua API
    },
  },
```

Atividade 8: Teste (TDD)



- Escrever dois casos de teste na **calculadoraApi** (dentro da pasta /aula7/Testes/):
  - “Deve retornar 5 ao dividir 15/3”
  - “Deve retornar erro ao dividir 15/0”
- Executar o teste, com **npx mocha**, e enviar evidências do resultado do teste
- Atualmente esses dois testes falham porque não foram implementados:

```
API Tests
  ✓ should return 200 OK

Testando operações
  ✓ Deve retornar 5 ao subtrair 8 - 3
  ✓ Deve retornar 5 ao somar 2 + 3
  1) Deve retornar 5 ao dividir 15 / 3
  2) Deve retornar erro ao dividir 15 / 0
  ✓ Deve retornar erro ao multiplicar 5 * 1


4 passing (4s)
2 failing
```

app.test.js

Testes > calculadoraApi > test > app.test.js > ...

```
35    });
36
37    it('Deve retornar 5 ao dividir 15 / 3', (done) => {
38      // Escrever código aqui
39    });
40
41    it('Deve retornar erro ao dividir 15 / 0', (done) => {
42      // Escrever código aqui
43    });
44
45    it('Deve retornar erro ao multiplicar 5 * 1', (done) => {
46      request(app)
47        .get('/divisao/15/3')
48        .expect(200)
49        .end((err, res) => {
50          if (err) return done(err);
51          expect(res.body.result).to.equal(5);
52          done();
53        });
54    });
55
56  });
57
```